

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	PPP
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	

```
UU      UU  EEEEEEEEE  TTTTTTTTT  NN      NN  EEEEEEEEE  TTTTTTTTT  SSSSSSSS  000000  000000
UU      UU  EEEEEEEEE  TTTTTTTTT  NN      NN  EEEEEEEEE  TTTTTTTTT  SSSSSSSS  000000  000000
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UU      UU  EE          TT          NNNN     NN  EE          TT          SS          00      00
UU      UU  EE          TT          NNNN     NN  EE          TT          SS          00      00
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UU      UU  EEEEEEEEE  TT          NN      NN  EEEEEEEEE  TT          SS          00      00
UU      UU  EEEEEEEEE  TT          NN      NN  EEEEEEEEE  TT          SS          00      00
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UU      UU  EE          TT          NN      NN  EE          TT          SS          00      00
UUUUUUUUUU  EEEEEEEEE  TT          NN      NN  EEEEEEEEE  TT          SSSSSSSS  000000  000000
UUUUUUUUUU  EEEEEEEEE  TT          NN      NN  EEEEEEEEE  TT          SSSSSSSS  000000  000000

LL      LL  IIIIIII  SSSSSSSS
LL      LL  IIIIIII  SSSSSSSS
LL      LL  II          SS
LL      LL  II          SS
LL      LL  II          SS
LL      LL  II          SS
LL      LL  II          SSSSSS
LL      LL  II          SSSSSS
LL      LL  II          SS
LL      LL  II          SS
LL      LL  II          SS
LL      LL  II          SS
LLLLLLLLLLL  IIIIIII  SSSSSSSS
LLLLLLLLLLL  IIIIIII  SSSSSSSS
```


(2)	68	Declarations
(3)	127	Read-Only Data
(4)	231	Read/Write Data
(5)	361	RMS-32 Data Structures
(6)	388	Main Program
(9)	563	NICE ROUTINE
(10)	750	System Service Exception Handler
(11)	889	RMS Error Handler
(12)	953	CTRL/C Handler
(13)	998	Error Exit
(14)	1054	Exit Handler

```

0000 1 .TITLE UETNETS00 VAX/VMS UETP checker for DECnet counters
0000 2 .IDENT 'V04-000'
0000 3 :
0000 4 :*****
0000 5 :
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :
0000 24 :
0000 25 :*****
0000 26 :
0000 27 :
0000 28 :++
0000 29 : FACILITY:
0000 30 : This module will be distributed with VAX/VMS under the [SYSTEST]
0000 31 : account.
0000 32 :
0000 33 : ABSTRACT:
0000 34 : This program will report all error indicating non-zero node and
0000 35 : circuit counters for all nodes and circuits indicated in the
0000 36 : UETININET.DAT file. If no counters indicate error then the node
0000 37 : name and circuit name will be reported with a success message.
0000 38 :
0000 39 : ENVIRONMENT:
0000 40 : This program will run in user access mode, with interrupts enabled
0000 41 : at all times. This program requires the following privileges and
0000 42 : quotas:
0000 43 : NETMBX
0000 44 :
0000 45 :--
0000 46 :
0000 47 : AUTHOR: Larry D. Jones, CREATION DATE: November, 1981
0000 48 :
0000 49 : MODIFIED BY:
0000 50 :
0000 51 : V03-004 RNH0002 Richard N. Holstein, 27-Mar-1983
0000 52 : Make use of new UETP error messages. Turn off System Service
0000 53 : failure exceptions when calling NML$INITIALIZE; we get snagged
0000 54 : on a logical name that's used for debugging purposes only. Fix
0000 55 : miscellaneous bugs in System Service error handling.
0000 56 :
0000 57 : V03-003 RNH0001 Richard N. Holstein, 21-Nov-1983

```


UETNETS00
V04-000

I 14
VAX/VMS UETP checker for DECnet counters 16-SEP-1984 01:29:03 VAX/VMS Macro V04-00
5-SEP-1984 04:25:57 [UETP.SRC]UETNETS00.MAR;1

Page 2
(1)

0000	58 :	Change error message wording when checking NICE messages.
0000	59 :	
0000	60 :	V03-002 LDJ0002 Larry D. Jones, 24-Jan-1983
0000	61 :	Modified to conform to area network format.
0000	62 :	
0000	63 :	V03-001 LDJ0001 Larry D. Jones, 24-Dec-1981
0000	64 :	Fixed missing node name access violation bug.
0000	65 :	
0000	66 : **	

```
0000 68      .SBTTL  Declarations
0000 69      :
0000 70      INCLUDE FILES:
0000 71      :
0000 72      .LIBRARY /SHRLIB$:NMALIBRY.MLB/
0000 73      :
0000 74      :
0000 75      MACROS:
0000 76      :
0000 77      $CHFDEF      ; Condition handler frame definitions
0000 78      $DIBDEF      ; Device information block definitions
0000 79      $NMADEF      ; DECnet definitions
0000 80      $SHRDEF      ; Shared messages
0000 81      $STSDEF      ; Status return
0000 82      $UETPDEF     ; UETP
0000 83      :
0000 84      .MACRO  TBL_ENT ENT,VALUE,STRING
0000 85      .PC1...
0000 86      .WORD    <ENT@15>!VALUE
0000 87      .ADDRESS PC2...
0000 88      PC1...=.
0000 89      .PC2...
0000 90      .ASCIC    /STRING/
0000 91      PC2...=.
0000 92      .ENDM    TBL_ENT
0000 93      :
0000 94      EQUATED SYMBOLS:
0000 95      :
0000 96      Facility number definitions:
00000001 0000 97      RMS$K_FACILITY = 1
0000 98      :
0000 99      SHR message definitions:
00740000 0000 100      UETP = UETP$_FACILITY@ST$SV_FAC_NO ; Define the UETP facility code
0000 101      :
007410E0 0000 102      UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038 0000 103      UETP$_BEGINDD = UETP!SHR$_BEGINDD
00741080 0000 104      UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098 0000 105      UETP$_OPENIN = UETP!SHR$_OPENIN
00741130 0000 106      UETP$_TEXT = UETP!SHR$_TEXT
0000 107      :
0000 108      Internal flag bits...:
00000000 0000 109      SHRT RPRTV = 0 ; Set if short report format desired
00000001 0000 110      CONTROL_CV = 1 ; Set if CTRL/C AST received
00000002 0000 111      CIR_CNT-BADV = 2 ; Set if a bad circuit counter was detected
00000003 0000 112      NOD_CNT-BADV = 3 ; Set if a bad node counter was detected
00000006 0000 113      BEGIN_MSGV = 6 ; Set when "begin" msg has been output
0000 114      :
00000001 0000 115      ...and corresponding masks:
00000002 0000 116      SHRT RPRTM = 1@SHRT RPRTV
00000004 0000 117      CONTROL_CM = 1@CONTROL_CV
00000008 0000 118      CIR_CNT-BADM = 1@CIR_CNT-BADV
00000040 0000 119      NOD_CNT-BADM = 1@NOD_CNT-BADV
00000080 0000 120      BEGIN_MSGM = 1@BEGIN_MSGV
0000 121      :
00000084 0000 122      BIT7M = ^X80
00000000 0000 123      Miscellany:
00000001 0000 124      TEXT_BUFFER = 132 ; Internal text buffer size
00000000 0000      NOD = 0 ; Node ID constant
00000001 0000      CIR = 1 ; Circuit ID constant
```


UETNETS00
V04-000

VAX/VMS UETP checker for DECnet counters K 14
Declarations 16-SEP-1984 01:29:03 VAX/VMS Macro V04-00
5-SEP-1984 04:25:57 [UETP.SRC]UETNETS00.MAR;1

Page 4
(2)

0000001A 0000 125

TBL_SIZE = 26

; Network counter table size

```
0000 127 .SBTTL Read-Only Data
00000000 128 .PSECT RODATA,NOEXE,NOWRT,PAGE
0000 129
53 45 54 53 59 53 00000008'010E0000' 0000 130 ACNT_NAME: ; Process name on exit
54 000E 131 .ASCID /SYSTEST/
000F 132
54 45 4E 54 45 55 00000017'010E0000' 000F 133 TSTNAM: ; This test name
30 30 53 001D 134 .ASCID /UETNETS00/
0020 135
0020 136 NO_RMS_AST_TABLE: ; List of errors for which...
00000000' 0020 137 .LONG RMSS_BLN ; ...RMS cannot deliver an AST...
00000000' 0024 138 .LONG RMSS_BUSY ; ...even if one has an ERR= arg
00000000' 0028 139 .LONG RMSS_CDA ; Note that we can search table...
00000000' 002C 140 .LONG RMSS_FAB ; ...via MATCHC since <31:16>...
00000000' 0030 141 .LONG RMSS_RAB ; ...pattern can't be in <15:0>
00000014 0034 142 NRAT_LENGTH = .-NO_RMS_AST_TABLE
0034 143
45 44 4F 4D 0000003C'010E0000' 0034 144 MODE: ; Run mode logical name
0040 145 .ASCID /MODE/
0040 146
0000 003F 0040 147 TTNAME_ROPTR:
0000000A' 0044 148 .WORD 63,0
0048 149 .ADDRESS TTNAME
0048 150
47 4F 4C 2E 0048 151 LOGEXT: ; Log file extention
0048 152 .ASCII /.LOG/
004C 153
004C 154
65 74 72 6F 62 41 00000054'010E0000' 004C 155 CNTRLMSG:
72 65 73 75 20 61 20 61 69 76 20 64 005A 156 .ASCID \Aborted via a user CTRL/C\
43 2F 4C 52 54 43 20 0066
006D 157
006D 158
65 6C 69 66 00000075'010E0000' 006D 159 FILE: ; Fills in RMS_ERR_STRING
0079 160 .ASCID /file/
0079 161
64 72 6F 63 65 72 00000081'010E0000' 0079 162 RECORD: ; Fills in RMS_ERR_STRING
0087 163 .ASCID /record/
0087 164
41 21 20 53 4D 52 0000008F'010E0000' 0087 165 RMS_ERR_STRING: ; Announces an RMS error
66 20 6E 69 20 72 6F 72 72 65 20 53 0095 166 .ASCID /RMS !AS error in file !AD/
44 41 21 20 65 6C 69 00A1
00A8 167
00A8 168 NMLINIT_ERR:
6F 77 74 65 6E 20 67 6E 69 72 75 64 00B6 169 .ASCID /Error during network communications initialization./
61 63 69 6E 75 6D 6D 6F 63 20 6B 72 00C2
61 69 74 69 6E 69 20 73 6E 6F 69 74 00CE
2E 6E 6F 69 74 61 7A 69 6C 00DA
00E3 170
00E3 171 ERR_MSG_CTR:
72 20 45 43 49 4E 000000EB'010E0000' 00E3 172 .ASCID /NICE response error code !XB, error message: !AC./
6F 72 72 65 20 65 73 6E 6F 70 73 65 00F1
```



```
20 2C 42 58 21 20 65 64 6F 63 20 72 00FD
67 61 73 73 65 6D 20 72 6F 72 72 65 0109
2E 43 41 21 20 3A 65 0115
011C 173
011C 174 COUNTER_MSG:
011C 175 .ASCID /!AC !AC !AC !AC !AC = !UL./
012A
0136
013E 176
013E 177 NODE:
013E 178 .ASCIC /Node /
0144 179
0144 180 CIRCUIT:
0144 181 .ASCIC /Circuit/
014C 182
014C 183 TO:
014C 184 .ASCIC /to/
014F 185
014F 186 THRU:
014F 187 .ASCIC /over/
0154 188
0154 189 CASE_FAILED:
0154 190 .ASCID /Unrecognized counter in NICÉ message./
0162
016E
017A
0181 191
0181 192 CIRCUIT_OK:
0181 193 .ASCID /Circuit !AC to !AC OK./
018F
019B
019F 194
019F 195 ZERO:
019F 196 .LONG 0
01A3 197 CNTR_TBL:
01A3 198 PC1... =
01A3 199 .+. <TBL_SIZE*6>
023F 200 TBL_END:
023F 201 PC2... =
023F 202 .LIST MEB
023F 203 TBL_ENT CIR,NMASC_CTCIR_ACL,<arriving congestion loss>
023F .=PC1...
023F .WORD <CIR@15>!NMASC_CTCIR_ACL
023F .ADDRESS PC2...
023F .=PC2...
023F .ASCIC /arriving congestion loss/
024B
0257
025F 18
0258 204 .NLIST MEB
0258 205 TBL_ENT CIR,NMASC_CTCIR_CRL,<corruption loss>
0268 206 TBL_ENT CIR,NMASC_CTCIR_TCL,<transit congestion loss>
0280 207 TBL_ENT CIR,NMASC_CTCIR_LDN,<line down>
```

028A	208	TBL_ENT CIR,NMASC-CTCIR-IFL,<initialization failure>
02A1	209	TBL_ENT CIR,NMASC-CTCIR-DEI,<data errors inbound>
02B5	210	TBL_ENT CIR,NMASC-CTCIR-DEO,<data errors outbound>
02CA	211	TBL_ENT CIR,NMASC-CTCIR-RRT,<remote reply timeouts>
02E0	212	TBL_ENT CIR,NMASC-CTCIR-LRT,<local reply timeouts>
02F5	213	TBL_ENT CIR,NMASC-CTCIR-RBE,<remote buffer errors>
030A	214	TBL_ENT CIR,NMASC-CTCIR-LBE,<local buffer errors>
031E	215	TBL_ENT CIR,NMASC-CTCIR-SLT,<selection timeouts>
0331	216	TBL_ENT CIR,NMASC-CTCIR-RPE,<remote process errors>
0347	217	TBL_ENT CIR,NMASC-CTCIR-LPE,<local process errors>
035C	218	TBL_ENT CIR,NMASC-CTCIR-LIR,<locally initiated resets>
0375	219	TBL_ENT CIR,NMASC-CTCIR-RIR,<remotely initiated resets>
038F	220	TBL_ENT CIR,NMASC-CTCIR-NIR,<network initiated resets>
03A8	221	TBL_ENT NOD,NMASC-CTNOD-RTO,<response timeouts>
03BA	222	TBL_ENT NOD,NMASC-CTNOD-RSE,<received connect resource errors>
03DB	223	TBL_ENT NOD,NMASC-CTNOD-APL,<aged packet loss>
03EC	224	TBL_ENT NOD,NMASC-CTNOD-NUL,<node unreachable packet loss>
0409	225	TBL_ENT NOD,NMASC-CTNOD-NOL,<node out of range packet loss>
0427	226	TBL_ENT NOD,NMASC-CTNOD-OPL,<oversized packet loss>
043D	227	TBL_ENT NOD,NMASC-CTNOD-PFE,<packet format error>
0451	228	TBL_ENT NOD,NMASC-CTNOD-RUL,<partial routing update loss>
046D	229	TBL_ENT NOD,NMASC-CTNOD-VER,<verification reject>


```
00000000 0481 231 .SBTTL Read/Write Data
00000000 232 .PSECT RWDATA,WRT,NOEXE,PAGE
0000 233
0000 234 TTCHAN: ; Channel associated with ctrl. term.
0000 235 .WORD 0
0002 236
0002 237 TTNAME_RWPTR:
0002 238 .WORD TTNAME_LEN,0
0000000A 0006 239 .ADDRESS TTNAME
000A 240 TTNAME:
000A 241 .ASCII /SYS$COMMAND/
0000000B 0015 242 TTNAME_LEN = .-TTNAME
00000049 0015 243 .BLKB 63-TTNAME_LEN
0049 244
0049 245
0049 246 FLAG: ; Miscellaneous flag bits
0000 0049 247 .WORD 0 ; (See Equated Symbols for definitions)
004B 248
004B 249 DEV: ; Device Information Block
00000074 004B 250 .LONG DIB$K_LENGTH
00000053 004F 251 .ADDRESS DEVBUF
0053 252 DEVBUF: ; Device Information Block
000000C7 0053 253 .BLKB DIB$K_LENGTH
00C7 254
00C7 255 FAO_BUF: ; FAO output string descriptor
0000 0084 00C7 256 .WORD TEXT_BUFFER,0
000000D7 00CB 257 .ADDRESS BUFFER
00CF 258
00CF 259 BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 0084 00CF 260 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
000000D7 00D3 261 .ADDRESS BUFFER
00D7 262
00D7 263 BUFFER: ; FAO output and other misc. buffer
0000015B 00D7 264 .BLKB TEXT_BUFFER
015B 265
015B 266 ERROR_COUNT: ; Cumulative error count at runtime
00000000 015B 267 .LONG 0
015F 268
015F 269 STATUS: ; Status value on program exit
00000000 015F 270 .LONG 0
0163 271
0163 272
0163 273 MSG_BLOCK: ; Auxiliary $GETMSG info
00000167 0163 274 .BLKB 4
0167 275
0167 276 EXIT_DESC: ; Exit handler descriptor
00000000 0167 277 .LONG 0
0000071C 016B 278 .ADDRESS EXIT_HANDLER
00000001 016F 279 .LONG 1
0000015F 0173 280 .ADDRESS STATUS
0177 281
0177 282 ARG_COUNT: ; Argument counter used by ERROR_EXIT
00000000 0177 283 .LONG 0
017B 284
017B 285 AREA_ADR_DESC:
00000000 017B 286 .LONG 0
00000000 017F 287 .ADDRESS 0
```



```
0183 288
0183 289 NODE_ADR_DESC:
00000000 0183 290 .LONG 0
00000000 0187 291 .ADDRESS 0
018B 292
018B 293 NICE_MSG:
00000005 018B 294 .LONG NICE_SIZE
0000019E 018F 295 .ADDRESS NICE_MESSAGE
0193 296
0193 297 NICE1_MSG:
00000002 0193 298 .LONG NICE1_SIZE
000001A3 0197 299 .ADDRESS NICET_MESSAGE
019B 300
019B 301 .ALIGN LONG
019C 302
0000 019C 303 AREA_WRD: ; Network area number
019C 304 .WORD 0
019E 305
019E 306 ;
019E 307 ; *** Warning ***
019E 308 ; The following section of data must remain contiguous.
019E 309 ;
019E 310 ; NICE packets used to get the counters.
019E 311 ;
019E 312
019E 313 NICE_MESSAGE:
14 019E 314 .BYTE NMASC_FNC_REA ; Read information function code
30 019F 315 .BYTE NMASC_OPINF_COU@NMASV_OPT_INF ; OPTION = Node, Counters, Volatile
00 01A0 316 .BYTE NMASC_ENT_NOD ; Node format = node address
01A1 317 NODE_WRD:
0000 01A1 318 .WORD 0
00000005 01A3 319 NICE_SIZE = .-NICE_MESSAGE
01A3 320
01A3 321 NICE1_MESSAGE:
14 01A3 322 .BYTE NMASC_FNC_REA ; Read information function code
33 01A4 323 .BYTE <<NMASC_OPINF_COU@NMASV_OPT_INF>>!-- ; OPTION = Circuit, Counters, Volatile
01A5 324 <NMASC_ENT_CIR>>
00000002 01A5 325 NICE1_SIZE = .-NICET_MESSAGE
01A5 326
01A5 327 CIRC_NAME:
000001AF 01A5 328 .BLKB 10
01AF 329
01AF 330 ;
01AF 331 ; *** End of warning ***
01AF 332 ;
01AF 333
01AF 334 NODE_NAME:
000001B6 01AF 335 .BLKB 7
01B6 336
01B6 337 AREA_ADR:
000001B9 01B6 338 .BLKB 3
01B9 339
01B9 340 NODE_ADR:
000001BE 01B9 341 .BLKB 5
01BE 342
01BE 343 NAME:
000001D7 01BE 344 .BLKB 25
```


UETNETS00
V04-000

VAX/VMS UETP checker for DECnet counters
Read/Write Data

D 15

16-SEP-1984 01:29:03
5-SEP-1984 04:25:57

VAX/VMS Macro V04-00
[UETP.SRC]UETNETS00.MAR;1

Page 10
(4)

	01D7	345		
	01D7	346	COUNTER:	
00000000	01D7	347	.LONG	0
	01DB	348		
	01DB	349	TYPE:	
00000000	01DB	350	.LONG	0
	01DF	351		
	01DF	352	TYPE1:	
00000000	01DF	353	.LONG	0
	01E3	354		
	01E3	355	TYPE2:	
00000000	01E3	356	.LONG	0
	01E7	357		
	01E7	358	END_ADR:	
00000000	01E7	359	.LONG	0

```

01EB 361      .SBTTL RMS-32 Data Structures
01EB 362      .ALIGN LONG
01EC 363
01EC 364  INI_FAB:      ; Allocate FAB for UETININET
01EC 365      $FAB-
01EC 366      FAC = GET,-
01EC 367      RAT = CR,-
01EC 368      SHR = GET,-
01EC 369      FNM = <UETININET.DAT>
023C 370
023C 371  INI_RAB:      ; Allocate RAB for UETININET
023C 372      $RAB-
023C 373      FAB = INI_FAB,-
023C 374      UBF = BUFFER,-
023C 375      USZ = TEXT_BUFFER,-
023C 376      RBF = BUFFER
0280 377
0280 378
0280 379  LOG_FAB:      ; Log file FAB
0280 380      $FAB      FNM = <UETNETS00.LOG>,-
0280 381      RAT = CR,-
0280 382      FAC = PUT
02D0 383  LOG_RAB:      ; Log file RAB
02D0 384      $RAB      FAB = LOG_FAB,-
02D0 385      RBF = BUFFER,-
02D0 386      RSZ = TEXT_BUFFER

```



```
0314 388 .SBTTL Main Program
00000000 389 .PSECT UETNETS00,EXE,NOWRT,PAGE
0000 390
0000 391 .DEFAULT DISPLACEMENT,WORD
0000 392
0000 393 .ENTRY UETNETS00,^M<> ; Entry mask
6D 04F7'CF DE 0002 394
0002 395 MOVAL SSERROR,(FP) ; Declare exception handler
0007 396 $SETSFM_S ENBFLG = #1 ; Enable system service failure mode
0010 397 $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
001B 398 $CREATE FAB = LOG_FAB,-
001B 399 ERR = RMS_ERROR ; Create the log file
002A 400 $CONNECT RAB = LOG_RAB,-
002A 401 ERR = RMS_ERROR ; Connect the RAB
0039 402 $OPEN FAB = INI_FAB,-
0039 403 ERR = RMS_ERROR ; Open the UETININET.DAT file
0048 404 $CONNECT RAB = INI_RAB,-
0048 405 ERR = RMS_ERROR ; Connect the RAB
7E D4 0057 406 CLRL -(SP) ; Set the time stamp flag
000F'CF DF 0059 407 PUSHAL TSTNAM ; Set the test name
02 DD 005D 408 PUSHL #2 ; Push the argument count
00741039 8F DD 005F 409 PUSHL #UETPS_BEGIN!STSSK_SUCCESS ; Set the message code
00000000'GF 04 FB 0065 410 CALLS #4,G^LIB$SIGNAL ; Print the startup message
0049'CF 0040 8F A8 006C 411 BISW2 #BEGIN MSGM,FLAG ; Set flag so we don't type it twice
0073 412 $SETPRN_S PRCNAM = TSTNAM ; Set the process name
007E 413 10$:
007E 414 $TRNLOG_S LOGNAM = TTNAME_RWPTR,-
007E 415 RSLLEN = TTNAME_RWPTR,-
007E 416 RSLBUF = TTNAME_ROPTR ; Translate the logical name
0006'CF 000A'CF DE 0097 417 MOVAL TTNAME,TTNAME_RWPTR+4 ; Undo possible previous PPF fixup
00000000'8F 50 D1 009E 418 CMPL R0,#SS$_NOTRAN ; Have we reached the end yet?
13 13 00A5 419 BEQL 20$ ; Br if yes
000A'CF 1B 91 00A7 420 CMPB #^X1B,TTNAME ; Is this a process permanent file?
D0 12 00AC 421 BNEQ 10$ ; Br if not
0002'CF 04 A2 00AE 422 SUBW #4,TTNAME_RWPTR ; Remove RMS overhead from PPF name...
0006'CF 04 C0 00B3 423 ADDL #4,TTNAME_RWPTR+4 ;
C4 11 00B8 424 BRB 10$ ; Now it's safe to retranslate
00BA 425 20$:
00BA 426 $GETDEV_S DEVNAM = TTNAME_RWPTR,-
00BA 427 PRIBUF = DEV ; Get its device type
00'8F 0057'CF 91 00CF 428 CMPB DEVBUF+DIB$_DEVCLASS,#DC$_TERM ; Is this a terminal?
45 12 00D5 429 BNEQ 30$ ; BR if no
00D7 430 $ASSIGN_S DEVNAM = TTNAME_RWPTR,- ; Set up for CTRL/C AST's
00D7 431 CHAN = TTCHAN
00E8 432 $QIOW_S CHAN = TTCHAN,- ; Enable CTRL/C AST's...
00E8 433 FUNC = #IOS$_SETMODE!IOS$_CTRLCAST,-
00E8 434 P1 = CCASTHAND
000F'CF DF 0109 435 PUSHAL TSTNAM ; ...and tell the user...
01 DD 010D 436 PUSHL #1
0074832B 8F DD 010F 437 PUSHL #UETPS_ABORT!STSSK_SUCCESS ; ...how to abort gracefully...
00000000'GF 03 FB 0115 438 CALLS #3,G^LIB$SIGNAL ; ...
011C 439 30$:
011C 440 $SETSFM_S ENBFLG = #0 ; While initializing net comm stuff...
00000000'GF 00 FB 0125 441 CALLS #0,G^NML$INITIALIZE
50 DD 012C 442 PUSHL R0
012E 443 $SETSFM_S ENBFLG = #1 ; ...don't die for lack of logical names
015F'CF 8ED0 0137 444 POPL STATUS
```

UETNETS00
V04-000

VAX/VMS UETP checker for DECnet counters 16-SEP-1984 01:29:03 VAX/VMS Macro V04-00
Main Program 5-SEP-1984 04:25:57 [UETP.SRC]UETNETS00.MAR;1

Page 13
(6)

```
11 015F'CF E8 013C 445 BLBS STATUS_LOOP ; BR if we initialized correctly
    00A8'CF DF 0141 446 PUSHAL NMLINIT_ERR
      01 DD 0145 447 PUSHL #1
00741132 8F DD 0147 448 PUSHL #UETPS_TEXT!STSSK_ERROR
      04 DD 014D 449 PUSHL #4
    0561 31 014F 450 BRW ERROR_EXIT
      0152 451
      0152 452 ; Fall into main processing loop.
```



```
00D7'CF 20444E45 8F D1 0152 454 LOOP:
                                0152 455
                                0152 456
                                0161 457
                                016A 458
                                016C 459
                                016F 460 10$:
56 51 000000D7'CF 3A 016F 461
01A6'CF 57 51 01A5'CF 56 C3 0174 462
59 00D7'CF 56 0177 463
025E'CF 59 01 017F 464
59 0184 465
0188 466
0190 467
0196 468
0199 469
019B 470
019F 471
01A1 472
01A5 473
01A8 474
01AA 475
01AF 476
01B2 477
01B8 478
01BD 479
01C2 480
01C4 481
01C8 482
01CC 483
01D3 484
01D7 485 20$:
67 59 20 3A 01D7 486
56 51 57 C3 01DB 487
01B9'CF 56 90 01DF 488
58 51 D0 01E4 489
01BA'CF 67 56 28 01E7 490
0183'CF 56 9B 01ED 491
0187'CF 67 DE 01F2 492
02 01F7 493
01A1'CF DF 01F9 494
0183'CF DF 01FD 495
00000000'GF 03 FB 0201 496
01A1'CF 56 06 0A 019C'CF F0 0208 497
58 000000D7'8F C3 0211 498
01AF'CF 94 0219 499
56 025E'CF 56 A3 021D 500
17 13 0223 501
01AF'CF 56 90 0225 502
01AF'CF 97 022A 503
56 97 022E 504
58 51 01 C1 0230 505
01B0'CF 68 56 28 0234 506
30 11 023A 507
023C 508 25$:
53 01B0'CF 5B D4 023C 509
DE 023E 510

$GET RAB = INI_RAB,- ; Get a record
ERR = RMS_ERROR ;
#^A/END /,BUFFER ; End of the file?
10$ BR if not
BRW SUC_EXIT ; Else end

LOCC #^A/ /,INI_RAB+RAB$W_RSZ,-
BUFFER ; Find the end of the circuit name
SUBL3 #BUFFER,R1,R6 ; Get circuit name size
MOVB R6,CIRC_NAME ; Set circuit name length
ADDL3 #1,R1,R7 ; Set start of node address
MOVC3 R6,BUFFER,CIRC_NAME+1 ; Save circuit name
SUBW3 R6,INI_RAB+RAB$W_RSZ,R9 ; Get remaining string length
MOVZWL R9,R9 ; Clean the high word out
DECB R9 ; Get back to end of string
LOCC #^A/./,R9,(R7) ; Area number present?
BEQL 20$ BR if not
SUBL3 R7,R1,R6 ; Get area adr length
SUBL2 R6,R9 ; Update remaining string length to...
DECL R9 ; ...node number and beyond
MOVB R6,AREA_ADR ; Set area adr length
MOVL R1,R8 ; Save end of string address
MOVC3 R6,(R7),AREA_ADR+1 ; Save area adr
MOVZBW R6,AREA_ADR_DESC ; Init area address descriptor
MOVAL (R7),AREA_ADR_DESC+4
PUSHL #2 ; Two byte number returned
PUSHAL AREA_WRD ; Address of output
PUSHAL AREA_ADR_DESC ; Address of input
CALLS #3,G*OTSS$CVT_TI_L ; Make the string a word value
ADDL3 R1,#1,R7 ; Update the node address pointer

LOCC #^A/ /,R9,(R7) ; Find node adr end
SUBL3 R7,R1,R6 ; Get node adr length
MOVB R6,NODE_ADR ; Set node adr length
MOVL R1,R8 ; Save end of string address
MOVC3 R6,(R7),NODE_ADR+1 ; Save node adr
MOVZBW R6,NODE_ADR_DESC ; Init node address descriptor
MOVAL (R7),NODE_ADR_DESC+4
PUSHL #2 ; Two byte number returned
PUSHAL NODE_WRD ; Address of output
PUSHAL NODE_ADR_DESC ; Address of input
CALLS #3,G*OTSS$CVT_TI_L ; Make the string a word value
INSV AREA_WRD,#10,#6,NODE_WRD ; Save the area number in the node adr
SUBL3 #BUFFER,R8,R6 ; Calculate string size
CLRB NODE_NAME ; Clean out node name size
SUBW3 R6,INI_RAB+RAB$W_RSZ,R6
BEQL 25$ BR if no node name
MOVB R6,NODE_NAME ; Only use one byte worth of node name length
DECB NODE_NAME ; Correct node name size
DECB R6
ADDL3 #1,R1,R8 ; Set start of node name
MOVC3 R6,(R8),NODE_NAME+1 ; Save node name
BRB 30$

CLRL R11 ; Init size storage
MOVAL NODE_NAME+1,R3 ; Set initial destination address
```



```

      017B'CF 95 0243 511 TSTB AREA_ADR_DESC ; Is there an area number?
      11 13 0247 512 BEQL 27$ ; BR if not
      5B 017B'CF 9A 0249 513 MOVZBL AREA_ADR_DESC,R11 ; Get area string length
      5B D6 024E 514 INCL R11 ; Add one for the area decimal point
01B7'CF 017B'CF 2C 0250 515 MOVCS AREA_ADR_DESC,AREA_ADR+1,-
      63 5B 2E 0257 516 #^A/./,RT1,(R3) ; Save area number and decimal point
      025A 517 27$:
      5B 0183'CF C0 025A 518 ADDL2 NODE_ADR_DESC,R11 ; Add node number size
      01AF'CF 5B 90 025F 519 MOVB R11,NODE_NAME ; Store the size
01BA'CF 0183'CF 28 0264 520 MOVCS NODE_ADR_DESC,NODE_ADR+1,-
      63 026B 521 (R3) ; Save node number
      026C 522 30$:
      0324'CF DF 026C 523 PUSHAL NICE_ROUTINE ; Get the node counters
      018B'CF DF 0270 524 PUSHAL NICE_MSG
00000000'GF 02 FB 0274 525 CALLS #2,G^NML$PROCESS,NICE
      05 0049'CF 02 E1 027B 526 BBC #CIR_CNT_BADV,FLAG,40$ ; BR if no node counter data
      0049'CF 08 A3 0281 527 BISW2 #NOD_CNT_BADM,FLAG ; Save a copy of the flag
      0286 528 40$:
      0193'CF 03 D0 0286 529 MOVL #NICE1_SIZE+1,NICE1_MSG ; Calculate NICE packet size
0193'CF 01A5'CF 80 028B 530 ADDB2 CIRC_NAME,NICE1_MSG ; Add in circuit name size
      0324'CF DF 0292 531 PUSHAL NICE_ROUTINE ; Get the circuit counters
      0193'CF DF 0296 532 PUSHAL NICE1_MSG
00000000'GF 02 FB 029A 533 CALLS #2,G^NML$PROCESS,NICE
      4A 0049'CF 03 E4 02A1 534 BBSC #NOD_CNT_BADV,FLAG,60$ ; BR if counters found bad
      44 0049'CF 02 E4 02A7 535 BBSC #CIR_CNT_BADV,FLAG,60$ ; BR if counters found bad
01DB'CF 01AF'CF DE 02AD 536 MOVAL NODE_NAME,TYPE ; Save the node name address
      01AF'CF 95 02B4 537 TSTB NODE_NAME ; Anything there?
      07 12 02B8 538 BNEQ 50$ ; BR if yes else...
01DB'CF 01B9'CF DE 02BA 539 MOVAL NODE_ADR,TYPE ; ...use the node address
      02C1 540 50$:
      02C1 541 $FAO_S CTRSTR = CIRCUIT_OK,- ; Print the circuit OK message
      02C1 542 OUTLEN = BUFFER_PTR,-
      02C1 543 OUTBUF = FAO_BUF,-
      02C1 544 P1 = #NAME,-
      02C1 545 P2 = TYPE
      00CF'CF DF 02DE 546 PUSHAL BUFFER_PTR ; Push the string address
      01 DD 02E2 547 PUSHL #1 ; Push the parameter counter
      00741131 8F DD 02E4 548 PUSHL #UETP$.TEXT!STSSK_SUCCESS ; Push signal name
00000000'GF 03 FB 02EA 549 CALLS #3,G^LIB$SIGNAL ; Print circuit OK
      02F1 550 60$:
      FE5E 31 02F1 551 BRW LOOP ; Do the next record
```


00000000'GF	00	FB	02F4	553	SUC_EXIT:		
	00	DD	02F4	554	CALLS	#0,G^NML\$TERMINATE	; Terminate the NML session
000F'CF	02	DF	02FB	555	PUSHL	#0	; Set the time flag
	02	DD	02FD	556	PUSHAL	TSTNAM	; Push the test name
00741081	8F	DD	0301	557	PUSHL	#2	; Push arg count
00000000'GF	04	DD	0303	558	PUSHL	#UETPS_ENDEDD!ST\$K_SUCCESS	; Push signal name
015F'CF	10000000'8F	FB	0309	559	CALLS	#4,G^LIB\$SIGNAL	; Output the message
		DD	0310	560	MOVL	#\$\$\$ NORMAL!ST\$M_INHIB_MSG,STATUS	; Set successful exit status
			0319	561	SEEXIT_S	STATUS	; Exit with the status

```
0324 563 .SBTTL NICE_ROUTINE
0324 564 :++
0324 565 : FUNCTIONAL DESCRIPTION:
0324 566 : This routine is the NICE response servicing routine. All calls to
0324 567 : NML$PROCESS_NICE specify this routine as the action routine.
0324 568 :
0324 569 : CALLING SEQUENCE:
0324 570 : PUSHAL NICE_ROUTINE
0324 571 : PUSHAL NICE_MSG_DESC
0324 572 : CALLS #2, G^NML$PROCESS_NICE
0324 573 :
0324 574 : INPUT PARAMETERS:
0324 575 : 4(AP) = Address of a response message descriptor
0324 576 :
0324 577 : IMPLICIT INPUTS:
0324 578 : NONE
0324 579 :
0324 580 : OUTPUT PARAMETERS:
0324 581 : NONE
0324 582 :
0324 583 : IMPLICIT OUTPUTS:
0324 584 : Error or success messages
0324 585 :
0324 586 : COMPLETION CODES:
0324 587 : NONE
0324 588 :
0324 589 : SIDE EFFECTS:
0324 590 : NONE
0324 591 :
0324 592 :--
0324 593
0324 594 NICE_ROUTINE:
0324 595 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0324 596 MOVL 4(AP),R6 ; Get the descriptor address
0324 597 MOVZWL (R6),R7 ; Get the response size
0324 598 ADDL3 R7,4(R6),END_ADR ; Save the response end address
0324 599 MOVL 4(R6),R6 ; Get the response address
0324 600 MOVZBL (R6)+,R8 ; Get the status code
0324 601 CMPB R8,#NMASC_STS_MOR ; If this is a more message then...
0324 602 BNEQ 10$
0324 603 BRW NICE_EXIT ; ...exit
0324 604 10$:
0324 605 CMPB R8,#NMASC_STS_DON ; If this is a done message then...
0324 606 BNEQ 20$
0324 607 BRW NICE_EXIT ; ...exit
0324 608 20$:
0324 609 CMPB R8,#NMASC_STS_SUC ; If this is a success then...
0324 610 BEQL CHECK_IT ; ...process the response
0324 611 :
0324 612 : The nice response is in error and it is reported to the user.
0324 613 :
0324 614 MOVZBL -1(R6),R9 ; Get error code
0324 615 MOVAL 2(R6),R10 ; Get the error message address
0324 616 $FAO_S CTRSTR = ERR_MSG_CTR,-
0324 617 OUTLEN = BUFFER_PTR,-
0324 618 OUTBUF = FAO_BUF,-
0324 619 P1 = R9,-
```

01E7'CF

Address	Offset	Hex	Dec	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	Op466	Op467	Op468	Op469	Op470	Op471	Op472	Op473	Op474	Op475	Op476	Op477	Op478	Op479	Op480	Op481	Op482	Op483	Op484	Op485	Op486	Op487	Op488	Op489	Op490	Op491	Op492	Op493	Op494	Op495	Op496	Op497	Op498	Op499	Op500	Op501	Op502	Op503	Op504	Op505	Op506	Op507	Op508	Op509	Op510	Op511	Op512	Op513	Op514	Op515	Op516	Op517	Op518	Op519	Op520	Op521	Op522	Op523	Op524	Op525	Op526	Op527	Op528	Op529	Op530	Op531	Op532	Op533	Op534	Op535	Op536	Op537	Op538	Op539	Op540	Op541	Op542	Op543	Op544	Op545	Op546	Op547	Op548	Op549	Op550	Op551	Op552	Op553	Op554	Op555	Op556	Op557	Op558	Op559	Op560	Op561	Op562	Op563	Op564	Op565	Op566	Op567	Op568	Op569	Op570	Op571	Op572	Op573	Op574	Op575	Op576	Op577	Op578	Op579	Op580	Op581	Op582	Op583	Op584	Op585	Op586	Op587	Op588	Op589	Op590	Op591	Op592	Op593	Op594	Op595	Op596	Op597	Op598	Op599	Op600	Op601	Op602	Op603	Op604	Op605	Op606	Op607	Op608	Op609	Op610	Op611	Op612	Op613	Op614	Op615	Op616	Op617	Op618	Op619	Op620	Op621	Op622	Op623	Op624	Op625	Op626	Op627	Op628	Op629	Op630	Op631	Op632	Op633	Op634	Op635	Op636	Op637	Op638	Op639	Op640	Op641	Op642	Op643	Op644	Op645	Op646	Op647	Op648	Op649	Op650	Op651	Op652	Op653	Op654	Op655	Op656	Op657	Op658	Op659	Op660	Op661	Op662	Op663	Op664	Op665	Op666	Op667	Op668	Op669	Op670	Op671	Op672	Op673	Op674	Op675	Op676	Op677	Op678	Op679	Op680	Op681	Op682	Op683	Op684	Op685	Op686	Op687	Op688	Op689	Op690	Op691	Op692	Op693	Op694	Op695	Op696	Op697	Op698	Op699	Op700	Op701	Op702	Op703	Op704	Op705	Op706	Op707	Op708	Op709	Op710	Op711	Op712	Op713	Op714	Op715	Op716	Op717	Op718	Op719	Op720	Op721	Op722	Op723	Op724	Op725	Op726	Op727	Op728	Op729	Op730	Op731	Op732	Op733	Op734	Op735	Op736	Op737	Op738	Op739	Op740	Op741	Op742	Op743	Op744	Op745	Op746	Op747	Op748	Op749	Op750	Op751	Op752	Op753	Op754	Op755	Op756	Op757	Op758	Op759	Op760	Op761	Op762	Op763	Op764	Op765	Op766	Op767	Op768	Op769	Op770	Op771	Op772	Op773	Op774	Op775	Op776	Op777	Op778	Op779	Op780	Op781	Op782	Op783	Op784	Op785	Op786	Op787	Op788	Op789	Op790	Op791	Op792	Op793	Op794	Op795	Op796	Op797	Op798	Op799	Op800	Op801	Op802	Op803	Op804	Op805	Op806	Op807	Op808	Op809	Op810	Op811	Op812	Op813	Op814	Op815	Op816	Op817	Op818	Op819	Op820	Op821	Op822	Op823	Op824	Op825	Op826	Op827	Op828	Op829	Op830	Op831	Op832	Op833	Op834	Op835	Op836	Op837	Op838	Op839	Op840	Op841	Op842	Op843	Op844	Op845	Op846	Op847	Op848	Op849	Op850	Op851	Op852	Op853	Op854	Op855	Op856	Op857	Op858	Op859	Op860	Op861	Op862	Op863	Op864	Op865	Op866	Op867	Op868	Op869	Op870	Op871	Op872	Op873	Op874	Op875	Op876	Op877	Op878	Op879	Op880	Op881	Op882	Op883	Op884	Op885	Op886	Op887	Op888	Op889	Op890	Op891	Op892	Op893	Op894	Op895	Op896	Op897	Op898	Op899	Op900	Op901	Op902	Op903	Op904	Op905	Op906	Op907	Op908	Op909	Op910	Op911	Op912	Op913	Op914	Op915	Op916	Op917	Op918	Op919	Op920	Op921	Op922	Op923	Op924	Op925	Op926	Op927	Op928	Op929	Op930	Op931	Op932	Op933	Op934	Op935	Op936	Op937	Op938	Op939	Op940	Op941	Op942	Op943	Op944	Op945	Op946	Op947	Op948	Op949	Op950	Op951	Op952	Op953	Op954	Op955	Op956	Op957	Op958	Op959	Op960	Op961	Op962	Op963	Op964	Op965	Op966	Op967	Op968	Op969	Op970	Op971	Op972	Op973	Op974	Op975	Op976	Op977	Op978	Op979	Op980	Op981	Op982	Op983	Op984	Op985	Op986	Op987	Op988	Op989	Op990	Op991	Op992	Op993	Op994	Op995	Op996	Op997	Op998	Op999	Op1000	Op1001	Op1002	Op1003	Op1004	Op1005	Op1006	Op1007	Op1008	Op1009	Op1010	Op1011	Op1012	Op1013	Op1014	Op1015	Op1016	Op1017	Op1018	Op1019	Op1020	Op1021	Op1022	Op1023	Op1024	Op1025	Op1026	Op1027	Op1028	Op1029	Op1030	Op1031	Op1032	Op1033	Op1034	Op1035	Op1036	Op1037	Op1038	Op1039	Op1040	Op1041	Op1042	Op1043	Op1044	Op1045	Op1046	Op1047	Op1048	Op1049	Op1050	Op1051	Op1052	Op1053	Op1054	Op1055	Op1056	Op1057	Op1058	Op1059	Op1060	Op1061	Op1062	Op1063	Op1064	Op1065	Op1066	Op1067	Op1068	Op1069	Op1070	Op1071	Op1072	Op1073	Op1074	Op1075	Op1076	Op1077	Op1078	Op1079	Op1080	Op1081	Op1082	Op1083	Op1084	Op1085	Op1086	Op1087	Op1088	Op1089	Op1090	Op1091	Op1092	Op1093	Op1094	Op1095	Op1096	Op1097	Op1098	Op1099	Op1100	Op1101	Op1102	Op1103	Op1104	Op1105	Op1106	Op1107	Op1108	Op1109	Op1110	Op1111	Op1112	Op1113	Op1114	Op1115	Op1116	Op1117	Op1118	Op1119	Op1120	Op1121	Op1122	Op1123	Op1124	Op1125	Op1126	Op1127	Op1128	Op1129	Op1130	Op1131	Op1132	Op1133	Op1134	Op1135	Op1136	Op1137	Op1138	Op1139	Op1140	Op1141	Op1142	Op1143	Op1144	Op1145	Op1146	Op1147	Op1148	Op1149	Op1150	Op1151	Op1152	Op1153	Op1154	Op1155	Op1156	Op1157	Op1158	Op1159	Op1160	Op1161	Op1162	Op1163	Op1164	Op1165	Op1166	Op1167	Op1168	Op1169	Op1170	Op1171	Op1172	Op1173	Op1174	Op1175	Op1176	Op1177	Op1178	Op1179	Op1180	Op1181	Op1182	Op1183	Op1184	Op1185	Op1186	Op1187	Op1188	Op1189	Op1190	Op1191	Op1192	Op1193	Op1194	Op1195	Op1196	Op1197	Op1198	Op1199	Op1200	Op1201	Op1202	Op1203	Op1204	Op1205	Op1206	Op1207	Op1208	Op1209	Op1210	Op1211	Op1212	Op1213	Op1214	Op1215	Op1216	Op1217	Op1218	Op1219	Op1220	Op1221	Op1222	Op1223	Op1224	Op1225	Op1226	Op1227	Op1228
---------	--------	-----	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------


```
00CF'CF DF 0359 620 P2 = R10
01 DD 0370 621 PUSHAL BUFFER_PTR ; Push the string address
00741132 8F DD 0374 622 PUSHL #1 ; Push the argument count
00000000'GF 03 FB 0376 623 PUSHL #UETP$TEXT!STSSK_ERROR ; Push the signal name
0049'CF 04 A8 037C 624 CALLS #3, G^CIBSSIGNAL ; Print the error message
015F'CF 00000000'8F DO 0383 625 BISW2 #CIR_CNT BADM,FLAG ; Set failure bit
0162 31 0388 626 MOVL #SS$BADPARAM,STATUS ; Set return status failure code
0391 627 BRW NICE_EXIT ; Thats it
0394 628 CHECK_IT:
56 03 0394 629 ADDL2 #3,R6 ; Skip error stuff
59 01A3'CF 04 DE 0397 630 MOVAL CNTR_TBL,R9 ; Set table address
0049'CF 06 AA 039C 631 BICW2 #CIR_CNT BADM,FLAG ; Clear the bad flag
01A1'CF 03 B1 03A1 632 CMPW (R6),NODE_WRD ; Is this a node response?
56 02 C0 03A6 633 BNEQ 10$ ; BR if not...
57 66 9A 03AB 634 ADDL2 #2,R6 ; ...else skip the node address word
57 80 8F 8A 03AB 635 10$: MOVZBL (R6),R7 ; Get the size of the name
01BE'CF 66 57 D6 03AE 637 BICB2 #BIT7M,R7 ; Incase this is the executor node
56 57 C0 03B2 638 INCL R7 ; Add in the count byte
01E7'CF 56 D1 03B4 639 MOVCL R7,(R6),NAME ; Save the name
03 12 03BD 640 ADDL2 R7,R6 ; Skip the name
012F 31 03BD 641 CHK_LOOP: CMPL R6,END_ADR ; All done?
58 86 3C 03BD 642 BNEQ 10$ ; BR if not...
00 0C EF 03C2 643 BRW NICE_EXIT ; ...else bail out
58 58 03C7 644 10$: MOVZWL (R6)+,R8 ; Get cntrl desc
0A 89 3C 03CA 647 EXTZV #NMASV_CNT_TYP,- ;
00 0C 03CC 648 #NMASV_CNT_TYP,- ;
58 58 03CD 649 R8,R8 ; Get the counter type
0A 89 3C 03CF 650 20$: MOVZWL (R9)+,R10 ; Get a table code
00 0C EF 03D2 651 EXTZV #NMASV_CNT_TYP,- ;
58 5A 03D4 652 #NMASV_CNT_TYP,- ;
58 58 D1 03D5 653 R10,R11 ; Get the counter type
04B 13 03D7 654 CMPL R8,R11 ; Is this it?
59 04 C0 03DA 655 BEQL 80$ ; BR if yes
0000023F'8F 59 D1 03DC 656 ADDL2 #4,R9 ; Skip name pointer
59 01A3'CF 02 DE 03DF 657 CMPL R9,#TBL_END ; End of table?
00 0C EF 03E6 658 BNEQ 20$ ; BR if not
58 FE A6 0C E1 03E8 659 MOVAL CNTR_TBL,R9 ; Set table address
03 56 02 C0 03ED 660 EXTZV #NMASV_CNT_WID,- ;
02 01 58 8F 03EF 661 #NMASV_CNT_WID,- ;
0017' 03F0 662 -2(R6),R8 ; Get the counter width
001C' 03F3 663 BBC #NMASV_CNT_MAP,-2(R6),30$ ; If not a mapped counter then carry on el
0022' 03F8 664 ADDL2 #2,R6 ; ...skip the map word
0154'CF DF 03FB 665 30$: CASEB R8,#1,#2 ; Skip the counter
00741132 8F DD 03FF 666 40$: .WORD 50$-40$
03 56 02 C0 0401 667 .WORD 60$-40$
02 01 58 8F 0403 668 .WORD 70$-40$
0017' DF 0405 669 PUSHAL CASE_FAILED ; Push the string address
00741132 8F DD 0409 670 PUSHL #1 ; Push the argument count
03 DD 040B 671 PUSHL #UETP$TEXT!STSSK_ERROR ; Push the signal name
029D 31 0411 672 PUSHL #3 ; Push the argument count
0413 673 BRW ERROR_EXIT ; Thats it
```



```

      56      D6      0416      677 50$:
      FFA2      31      0416      678
      56      02      C0      0418      679
      FF9C      31      041B      680 60$:
      56      04      C0      041B      681
      FF96      31      041E      682
      56      04      C0      0421      683 70$:
      FF96      31      0421      684
      56      04      C0      0424      685
      FF96      31      0427      686 80$:
      0D      EF      0427      687
      02      02      EF      0429      688
      58      FE      A6      042A      689
      03      FE      A6      0C      E1      042D      690
      56      02      C0      0432      691
      02      01      58      8F      0435      692 90$:
      02      01      58      8F      0435      693
      02      01      58      8F      0439      694 100$:
      0017'      0439      695
      001F'      043B      696
      0027'      043D      697
      0154'CF      DF      043F      698
      01      DD      0443      699
      00741132 8F      DD      0445      700
      03      DD      0448      701
      0263      31      044D      702
      01D7'CF      86      9A      0450      703 110$:
      000D      31      0450      704
      01D7'CF      86      3C      0458      706 120$:
      0005      31      0458      707
      01D7'CF      86      D0      0460      709 130$:
      01D7'CF      86      D0      0460      710
      01D7'CF      86      D0      0465      711 140$:
      01D7'CF      86      D0      0465      712
      59      01A3'CF      DE      0467      713
      FF4E      31      046C      714
      0049'CF      04      A8      046F      715 150$:
      01DB'CF      013E'CF      DE      0474      716
      5A      01AF'CF      DE      047B      717
      6A      95      0480      718
      05      12      0482      719
      5A      01B9'CF      DE      0484      720
      5A      01B9'CF      DE      0484      721
      01DF'CF      014F'CF      DE      0489      722 160$:
      01E3'CF      01A5'CF      DE      0490      723
      18 FF A9      07      E1      0497      724
      01DB'CF      0144'CF      DE      049C      725
      01DF'CF      014C'CF      DE      04A3      726
      01E3'CF      5A      D0      04AA      727
      5A      01BE'CF      DE      04AF      728
      5A      01BE'CF      DE      04AF      729
      04B4      730 170$:
      04B4      731
      04B4      732
      04B4      733

      INCL      R6      ; Skip a byte counter
      BRW      CHK_LOOP
      ADDL2     #2,R6      ; Skip a word counter
      BRW      CHK_LOOP
      ADDL2     #4,R6      ; Skip a long word counter
      BRW      CHK_LOOP
      EXTZV     #NMA$V_CNT_WID,-
      #NMA$S_CNT_WID,-
      -2(R6),R8      ; Get the counter width
      BBC      #NMA$V_CNT_MAP,-2(R6),90$ ; If not a mapped counter then carry on el
      ADDL2     #2,R6      ; ...skip the map word
      CASEB     R8,#1,#2      ; Skip the counter
      .WORD     110$-100$
      .WORD     120$-100$
      .WORD     130$-100$
      PUSHAL    CASE_FAILED      ; Push the string address
      PUSHL     #1      ; Push the argument count
      PUSHL     #UETP$_TEXT!ST$K_ERROR ; Push the signal name
      PUSHL     #3      ; Push the argument count
      BRW      ERROR_EXIT      ; That's it
      MOVZBL    (R6)+,COUNTER      ; Get a byte counter
      BRW      140$
      MOVZWL    (R6)+,COUNTER      ; Get a word counter
      BRW      140$
      MOVL      (R6)+,COUNTER      ; Get a long word counter
      BNEQ      150$
      MOVAL     CNTR_TBL,R9      ; Reset the table address
      BRW      CHK_LOOP      ; BR if counter was zero
      BISW2     #CIR_CNT_BADM,FLAG ; Set the bad one flag
      MOVAL     NODE,TYPE      ; Set the default entity type of node
      MOVAL     NODE_NAME,R10 ; Save the node name address
      TSTB      (R10)      ; Anything there
      BNEQ      160$      ; BR if yes else...
      MOVAL     NODE_ADR,R10 ; ...use the node address
      MOVAL     THRU,TYPE1      ; Set for node THRU circuit format
      MOVAL     CIRC_NAME,TYPE2
      BBC      #7,-T(R9),170$ ; Check to see if we guessed right
      MOVAL     CIRCUIT,TYPE ; If not set type to circuit
      MOVAL     TO,TYPE1      ; Set up for circuit to node format
      MOVL      R10,TYPE2
      MOVAL     NAME,R10
      $FAO_S     CTRSTR = COUNTER_MSG,- ; Generate a bad counter message
      OUTLEN = BUFFER_PTR,-
      OUTBUF = FAO_BUF,-
```


		04B4	734	P1	= TYPE,-	
		04B4	735	P2	= R10,-	
		04B4	736	P3	= TYPE1,-	
		04B4	737	P4	= TYPE2,-	
		04B4	738	P5	= (R9)-	
		04B4	739	P6	= COUNTER	
00CF'CF	DF	04DB	740	PUSHAL	BUFFER_PTR	: Push the string address
01	DD	04DF	741	PUSHL	#1	: Push the argument count
00741133 8F	DD	04E1	742	PUSHL	#UETP\$ TEXT!ST\$SK_INFO	: Push the signal name
00000000'GF 03	FB	04E7	743	CALLS	#3, G^[IB\$SIGNAL	: Print the error message
59 01A3'CF	DE	04EE	744	MOVAL	CNTR_TBL,R9	: Reset the counter table pointer
FEC7	31	04F3	745	BRW	CHK_LOOP	: Thats it
		04F6	746			
		04F6	747	NICE_EXIT:		
04	04F6	748	RET			


```

04F7 750 .SBTTL System Service Exception Handler
04F7 751 :++
04F7 752 : FUNCTIONAL DESCRIPTION:
04F7 753 : This routine is executed if a system service or RMS error occurs or
04F7 754 : if a LIB$SIGNAL system service is used to output a message.
04F7 755 :
04F7 756 : CALLING SEQUENCE:
04F7 757 : Entered via an exception from the system
04F7 758 :
04F7 759 : INPUT PARAMETERS:
04F7 760 : ERROR_COUNT = previous cumulative error count
04F7 761 :
04F7 762 : AP ---->
04F7 763 :
04F7 764 : SIGNAL ARY PNT
04F7 765 :
04F7 766 : MECH ARY PNT
04F7 767 :
04F7 768 : 4
04F7 769 :
04F7 770 : ESTABLISH FP
04F7 771 :
04F7 772 : DEPTH
04F7 773 :
04F7 774 : R0
04F7 775 :
04F7 776 : R1
04F7 777 :
04F7 778 : N
04F7 779 :
04F7 780 : CONDITION NAME
04F7 781 :
04F7 782 : N-3 ADDITIONAL
04F7 783 : LONG WORD ARGS
04F7 784 :
04F7 785 : PC
04F7 786 :
04F7 787 : PSL
04F7 788 :
04F7 789 : IMPLICIT INPUTS:
04F7 790 : NONE
04F7 791 :
04F7 792 : OUTPUT PARAMETERS:
04F7 793 : NONE
04F7 794 :
04F7 795 : IMPLICIT OUTPUTS:
04F7 796 : NONE
04F7 797 :
04F7 798 : COMPLETION CODES:
04F7 799 : NONE
04F7 800 :
04F7 801 : SIDE EFFECTS:
04F7 802 : NONE
04F7 803 : --
04F7 804 :
04F7 805 : SSERROR:
OFFC 04F7 806 :.WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask

```

2	
SIGNAL ARY PNT	
MECH ARY PNT	
4	
ESTABLISH FP	
DEPTH	Mechanism Array
R0	
R1	
N	
CONDITION NAME	
N-3 ADDITIONAL LONG WORD ARGS	Signal Array
PC	
PSL	


```
04F9 807
04F9 808
50 01 DD 0502 809
00' D1 0504 810
02 13 0507 811
6E D4 0509 812
050B 813 10$:
050B 814
50 01 DD 0514 815
00' D1 0516 816
02 13 0519 817
6E D4 051B 818
051D 819 20$:
56 04 AC D0 051D 820
59 04 A6 7D 0521 821
10 ED 0525 822
0C 0527 823
00000074 8F 59 0528 824
16 12 052E 825
66 02 C2 0530 826
0533 827
0533 828
0533 829
25 11 0544 830
0546 831 30$:
59 00000000'8F D1 0546 832
36 12 054D 833
10 ED 054F 834
0C 0551 835
00000000'8F 5A 0552 836
2B 12 0558 837
5A F0000000 8F CA 055A 838
08 A6 04 39 0561 839
14 0565 840
0020'CF 0566 841
1A 13 0569 842
056B 843 40$:
01 BA 056B 844
056D 845
01 BA 0576 846
0578 847
50 00' D0 0581 848
04 0584 849
0585 850 50$:
015F'CF 59 D0 0585 851
58 D4 058A 852
59 00000000'8F D1 058C 853
38 12 0593 854
0595 855
0595 856
0595 857
0595 858
0595 859
0164'CF 95 05AC 860
16 13 05B0 861
00CF'CF DF 05B2 862
01 DD 05B6 863

$SETAST_S ENBFLG = #0 ; Disable AST delivery
PUSHL #1 ; Assume ASTs were enabled
CMPL S^#SS$_WASSET,R0 ; Were ASTs enabled?
BEQL 10$ ; BR if they were
CLRL (SP) ; Set ASTs to remain disabled

$SETSFM_S ENBFLG = #0 ; Disable SS failure mode
PUSHL #1 ; Assume SS failure mode was enabled
CMPL S^#SS$_WASSET,R0 ; Was SS failure mode enabled?
BEQL 20$ ; BR if it was
CLRL (SP) ; Set SS failure mode to remain off

MOVL CHF$_SIGARGLIST(AP),R6 ; Get the signal array pointer
MOVQ CHF$_SIG_NAME(R6),R9 ; Get NAME in R9 and ARG1 in R10
CMPZV #STSSV_FAC_NO,- ; Is this a message from LIB$SIGNAL?
#STSS$_FAC_NO,-
R9,#JETS$_FACILITY
BNEQ 30$ ; BR if this is not a UETP exception
SUBL2 #2,CHF$_SIG_ARGS(R6) ; Drop the PC and PSL
$PUTMSG_S MSGVEC = - ; Print the message
CHF$_SIG_ARGS(R6),-
ACTRTN = 80$

BRB 40$ ; Restore ASTs and SS fail mode

CMPL #SS$_SSFAIL,R9 ; RMS failures are SysSvc failures
BNEQ 50$ ; BR if this can't be an RMS failure
CMPZV #STSSV_FAC_NO,- ; Is it an RMS failure?
#STSS$_FAC_NO,-
R10,#RMS$_FACILITY
BNEQ 50$ ; BR if not
BICL2 #^XF0000000,R10 ; Strip control bits from status code
MATCHC #4,CHF$_SIG_ARG1(R6),- ; Is it an RMS failure for which...
#NRAT_LENGTH,-
NO_RMS_AST_TABLE
BEQL 50$ ; ...no AST can be delivered?
; BR if so - must give error here

POPR #^M<R0> ; Restore SS failure mode...
$SETSFM_S ENBFLG = R0 ;
POPR #^M<R0> ; Restore AST enable...
$SETAST_S ENBFLG = R0 ;
MOVL S^#SS$_NORMAL,R0 ; Supply a standard status for exit
RET ; Resume processing (or goto RMS_ERROR)

MOVL R9,STATUS ; Save the status
CLRL R8 ; Assume for now it's not SS failure
CMPL #SS$_SSFAIL,R9 ; But is it a System Service failure?
BNEQ 70$ ; BR if not - no special case message
$GETMSG_S MSGID = R10,- ; Get SS failure code associated text
MSGLEN = BUFFER_PTR,-
BUFADR = FAO_BUF,-
FLAGS = #14,-
OUTADR = MSG_BLOCK

TSTB MSG_BLOCK+1 ; Get FAO arg count for SS failure code
BEQL 60$ ; Don't use $GETMSG if no $FAO args...
PUSHAL BUFFER_PTR ; ...else build up...
PUSHL #1 ; ...a message describing...
```



```
00741130 8F DD 05B8 864 PUSHL #UETP$ TEXT ; ...why the System Service failed
00 5A FO 05BE 865 INSV R10,#STSSV SEVERITY,- ; Give the message...
6E 03 05C1 866 #STSS SEVERITY,(SP) ; ...the correct severity code
58 03 DO 05C3 867 MOVL #3,R8 ; Count the number of args we pushed
05 11 05C6 868 BRB 70$
5A DD 05C8 869 60$: PUSHL R10 ; Save SS failure code
58 01 DO 05CA 871 MOVL #1,R8 ; Count the number of args we pushed
05CD 872 70$: MULL3 #4,CHF$$_SIG_ARGS(R6),R7 ; Get arglist length in bytes
57 66 04 C5 05CD 873 SUBL2 R7,SP ; Save the current signal array...
6E 04 A6 57 C2 05D1 874 MOVCL R7,CHF$$_SIG_NAME(R6),(SP) ; ...on the stack
7E 66 58 C1 05D9 875 ADDL3 R8,CHF$$_SIG_ARGS(R6),-(SP) ; Push the current arg count
00D3 31 05DD 876 BRW ERROR_EXIT
05E0 877
05E0 878 80$: .WORD ^M<R2> ; PUTMSG action routine
52 04 AC 0004 05E0 880 MOVCL 4(AP),R2 ; Get the message descriptor address
02F2'CF 62 3C 05E2 881 MOVZWL (R2),LOG_RAB+RAB$$_RSZ ; Get the message size
02F8'CF 04 A2 DO 05E6 882 MOVCL 4(R2),LOG_RAB+RAB$$_RBF ; Set the message address
05F1 883 $PUT RAB = LOG_RAB,-
05F1 884 ERR = RMS_ERROR ; Write the log file
50 00000000'8F DO 0600 885 MOVCL #SS$_NORMAL,R0 ; Set the return status code
04 0607 887 RET
```



```
0608 889 .SBTTL RMS Error Handler
0608 890 :++
0608 891 : FUNCTIONAL DESCRIPTION:
0608 892 :   This routine handles error returns from RMS calls.
0608 893 :
0608 894 : CALLING SEQUENCE:
0608 895 :   Called by RMS when a file processing error is found.
0608 896 :
0608 897 : INPUT PARAMETERS:
0608 898 :   NONE
0608 899 :
0608 900 : IMPLICIT INPUTS:
0608 901 :   The FAB or RAB associated with the RMS call.
0608 902 :
0608 903 : OUTPUT PARAMETERS:
0608 904 :   NONE
0608 905 :
0608 906 : IMPLICIT OUTPUTS:
0608 907 :   Error message
0608 908 :
0608 909 : COMPLETION CODES:
0608 910 :   NONE
0608 911 :
0608 912 : SIDE EFFECTS:
0608 913 :   Program may exit, depending on severity of the error.
0608 914 :
0608 915 :--
0608 916
0608 917 RMS_ERROR:
0608 918 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
060A 919
168 04 AC DO 060A 920 MOVL 4(AP),R6 ; See whether we're dealing with...
168 66 03 91 060E 921 CMPB #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
168 16 12 0611 922 BNEQ 10$ ; BR if it's a RAB
168 006D'CF DE 0613 923 MOVAL FILE,R7 ; FAB-specific code: text string...
168 58 56 DO 0618 924 MOVL R6,R8 ; ...address of FAB...
168 0C A6 DD 061B 925 PUSHL FAB$STV(R6) ; ...STV field for error...
168 08 A6 DD 061E 926 PUSHL FAB$STS(R6) ; ...STS field for error...
168 08 A6 DO 0621 927 MOVL FAB$STS(R6),STATUS ; ...and save the error code
168 15 11 0627 928 BRB RMS_COMMON ; FAB and RAB share other code
168 0629 929 10$:
168 0079'CF DE 0629 930 MOVAL RECORD,R7 ; RAB-specific code: text string...
168 58 3C A6 DO 062E 931 MOVL RAB$FAB(R6),R8 ; ...address of associated FAB...
168 0C A6 DD 0632 932 PUSHL RAB$STV(R6) ; ...STV field for error...
168 08 A6 DD 0635 933 PUSHL RAB$STS(R6) ; ...STS field for error...
168 08 A6 DO 0638 934 MOVL RAB$STS(R6),STATUS ; ...and save the error code
168 063E 935 RMS_COMMON:
168 5A 34 A8 9A 063E 936 MOVZBL FAB$B_FNS(R8),R10 ; Get the file name size
168 0642 937 $FAO_S CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
168 0642 938 OUTLEN = BUFFER_PTR,-
168 0642 939 OUTBUF = FAO_BUF,-
168 0642 940 P1 = R7,-
168 0642 941 P2 = R10,-
168 0642 942 P3 = FAB$L_FNA(R8)
168 00CF'CF DF 065C 943 PUSHAL BUFFER_PTR ; ...and arguments for ERROR_EXIT...
168 01 DD 0660 944 PUSHL #1 ; ...
168 00741130 8F DD 0662 945 PUSHL #UETP$TEXT ; ...
```


59	00	EF	0668	946	EXTZV	#STSSV_SEVERITY,-	
	03		066A	947		#STSSS_SEVERITY,-	
	015F'CF		066B	948		STATUS,R9	
	6E	59	88	066F	BISB2	R9,(SP)	; ...get the severity code...
		05	DD	0672	PUSHL	#5	; ...and add it into the signal name
	003C	31	0674	951	BRW	ERROR_EXIT	; Current arg count


```
0677 953 .SBTTL CTRL/C Handler
0677 954 :++
0677 955 : FUNCTIONAL DESCRIPTION:
0677 956 : This routine handles CTRL/C AST's
0677 957 :
0677 958 : CALLING SEQUENCE:
0677 959 : Called via AST
0677 960 :
0677 961 : INPUT PARAMETERS:
0677 962 : NONE
0677 963 :
0677 964 : IMPLICIT INPUTS:
0677 965 : NONE
0677 966 :
0677 967 : OUTPUT PARAMETERS:
0677 968 : NONE
0677 969 :
0677 970 : IMPLICIT OUTPUTS:
0677 971 : NONE
0677 972 :
0677 973 : COMPLETION CODES:
0677 974 : NONE
0677 975 :
0677 976 : SIDE EFFECTS:
0677 977 : NONE
0677 978 :
0677 979 :--
0677 980
0677 981 CCASTHAND:
OFFC 0677 982 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
0679 983
004C'CF DF 0679 984 PUSHAL CNTRLCMSG ; Set message pointer
01 DD 067D 985 PUSHL #1 ; Set arg count
00741130 8F DD 067F 986 PUSHL #UETPS_TEXT!STSSK_WARNING ; Set signal name
00 DD 0685 987 PUSHL #0 ; Indicate an abnormal termination
000F'CF DF 0687 988 PUSHAL TSTNAM ; ...
02 DD 068B 989 PUSHL #2 ; ...
007410E0 8F DD 068D 990 PUSHL #UETPS_ABENDD!STSSK_WARNING ; ...
00000000'GF 07 FB 0693 991 CALLS #7,G^LIBSSIGNAL ; Output the message
0049'CF 02 A8 069A 992 BISW2 #CONTROL_CM,FLAG ; Set CTRL/C flag bit...
DO 069F 993 MOVL #<SS$ CONTROL_C8^C7- ; ...and exit status
06A0 994 !STSSK_WARNING-
06A0 995 !STSM_INHIB_MSG>,STATUS
015F'CF 10000000'8F 06A8 996 $EXIT_S STATUS ; Terminate program cleanly
```



```
06B3 998 .SBTTL Error Exit
06B3 999 :++
06B3 1000 : FUNCTIONAL DESCRIPTION:
06B3 1001 : This routine prints an error message and exits.
06B3 1002 :
06B3 1003 : CALLING SEQUENCE:
06B3 1004 : MOVx error status value,STATUS
06B3 1005 : PUSHx error specific information on the stack
06B3 1006 : PUSHL current argument count
06B3 1007 : BRW ERROR_EXIT
06B3 1008 :
06B3 1009 : INPUT PARAMETERS:
06B3 1010 : Arguments to LIB$SIGNAL, as above
06B3 1011 :
06B3 1012 : IMPLICIT INPUTS:
06B3 1013 : NONE
06B3 1014 :
06B3 1015 : OUTPUT PARAMETERS:
06B3 1016 : Message to SYS$OUTPUT and SYS$ERROR
06B3 1017 :
06B3 1018 : IMPLICIT OUTPUTS:
06B3 1019 : Program exit
06B3 1020 :
06B3 1021 : COMPLETION CODES:
06B3 1022 : NONE
06B3 1023 :
06B3 1024 : SIDE EFFECTS:
06B3 1025 : NONE
06B3 1026 :
06B3 1027 :--
06B3 1028 :
06B3 1029 ERROR_EXIT:
06B3 1030 :
06B3 1031 $SETAST_S ENBFLG = #0 ; ASTs can play havoc with messages
15 0049'CF 06 E0 06B3 1032 BBS #BEGIN_MSGV,FLAG,10$ ; BR if 'begin' msg has already been output
06B3 1033 CLRL -(SP) ; Set the time stamp flag
06B3 1034 PUSHAL TSTNAM ; Set the test name
06B3 1035 PUSHL #2 ; Push the argument count
06B3 1036 PUSHL #UETP$_BEGIN!ST$K_SUCCESS ; Set the message code
00000000'GF 04 FB 06D0 1037 CALLS #4,G^LIB$SIGNAL ; Print the startup message
06D7 1038 10$:
0177'CF 08 8E C1 06D7 1039 ADDL3 (SP)+,#8,ARG_COUNT ; Get total # args, pop partial count
015B'CF D6 06DD 1040 INCL ERROR_COUNT ; Keep running error count
06B3 1041 PUSHL #0 ; Push the time parameter
06B3 1042 PUSHAL TSTNAM ; Push test name...
06B3 1043 PUSHL #2 ; ...arg count...
06B3 1044 PUSHL #UETP$_ABEND!ST$K_ERROR ; ...and signal name
06B3 1045 PUSHL ERROR_COUNT ; Finish off arg list...
06B3 1046 PUSHAL TSTNAM
06B3 1047 PUSHL #2
06B3 1048 PUSHL #UETP$_ERBOXPROC!ST$K_ERROR ; ...for error box message
00000000'GF 0177'CF FB 06FF 1049 CALLS ARG_COUNT,G^LIB$SIGNAL ; Truly hitch
0708 1050
015F'CF 10000000 8F C8 0708 1051 BISL #ST$M_INHIB_MSG,STATUS ; Don't print messages twice!
0711 1052 $EXIT_S STATUS ; Exit in error
```



```

071C 1054 .SBTTL Exit Handler
071C 1055 :++
071C 1056 : FUNCTIONAL DESCRIPTION:
071C 1057 : This routine handles cleanup on exits.
071C 1058 :
071C 1059 : CALLING SEQUENCE:
071C 1060 : Invoked automatically by $EXIT System Service.
071C 1061 :
071C 1062 : INPUT PARAMETERS:
071C 1063 : Location STATUS contains the exit status, FLAG has synchronizing bits.
071C 1064 :
071C 1065 : IMPLICIT INPUTS:
071C 1066 : NONE
071C 1067 :
071C 1068 : OUTPUT PARAMETERS:
071C 1069 : NONE
071C 1070 :
071C 1071 : IMPLICIT OUTPUTS:
071C 1072 : Various files are de-accessed, the process name is reset, and any
071C 1073 : necessary synchronization with UETPDEV01 is carried out.
071C 1074 :
071C 1075 : COMPLETION CODES:
071C 1076 : NONE
071C 1077 :
071C 1078 : SIDE EFFECTS:
071C 1079 : NONE
071C 1080 :
071C 1081 :--
071C 1082 :
071C 1083 EXIT_HANDLER:
OFFC 071C 1084 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
071E 1085
071E 1086 $SETSFMS ENBFLG = #0 ; Turn off System Service failure mode
0727 1087 $SETASTS ENBFLG = #0 ; We're finished - no more ASTs
0730 1088 $DISCONNECT RAB = INI_RAB ; Disconnect the RAB from the FAB
073B 1089 $CLOSE FAB = INI_FAB ; Close the UETINIDEV.DAT file
0746 1090 $DISCONNECT RAB = LOG_RAB ; Disconnect the RAB from the FAB
0751 1091 $CLOSE FAB = LOG_FAB ; Close the UETNETS00.LOG file
075C 1092 $SETPRNS PRCNAM = ACNT_NAME ; Reset the process name
04 0767 1093 RET ; That's all folks!
0768 1094
0768 1095 .END UETNETS00

```


\$\$TAB	= 000002D0 R	03	FABSL_FNA	= 0000002C		
\$\$TABEND	= 00000314 R	03	FABSL_FOP	= 00000004		
\$\$TMP	= 00000000		FABSL_STS	= 00000008		
\$\$TMP1	= 00000001		FABSL_STV	= 0000000C		
\$\$TMP2	= 000000CF		FABSV_CHAN_MODE	= 00000002		
\$\$TMPX	= 0000000D R	04	FABSV_CR	= 00000001		
\$\$TMPX1	= 0000000D		FABSV_FILE_MODE	= 00000004		
\$\$T1	= 00000001		FABSV_GET	= 00000001		
\$\$T2	= 00000006		FABSV_LNM_MODE	= 00000000		
ACNT_NAME	00000000 R	02	FABSV_PUT	= 00000000		
AREA_ADR	000001B6 R	03	FABSW_GBC	= 00000048		
AREA_ADR_DESC	0000017B R	03	FAO_BUF	000000C7 R	03	
AREA_WRD	000C019C R	03	FILE	0000006D R	02	
ARG_COUNT	00000177 R	03	FLAG	00000049 R	03	
BEGIN_MSGM	= 00000040		INI_FAB	000001EC R	03	
BEGIN_MSGV	= 00000006		INI_RAB	0000023C R	03	
BIT7M	= 00000080		IOSM_CTRLCAST	***** X	05	
BUFFER	000000D7 R	03	IOS_SETMODE	***** X	05	
BUFFER_PTR	000000CF R	03	LIBSSIGNAL	***** X	05	
CASE_FAILED	00000154 R	02	LOGEXT	00000048 R	02	
CCASTHAND	00000677 R	05	LOG_FAB	00000280 R	03	
CHECK_IT	000C0394 R	05	LOG_RAB	000002D0 R	03	
CHFSL_SIGARGLST	= 00000004		LOOP	00000152 R	05	
CHFSL_SIG_ARG1	= 00000008		MODE	00000034 R	02	
CHFSL_SIG_ARGS	= 00000000		MSG_BLOCK	00000163 R	03	
CHFSL_SIG_NAME	= 00000004		NAME	000001BE R	03	
CHK_LOOP	000003BD R	05	NICE1_MESSAGE	000001A3 R	03	
CIR	= 00000001		NICE1_MSG	00000193 R	03	
CIRCUIT	00000144 R	02	NICE1_SIZE	= 00000002		
CIRCUIT_OK	00000181 R	02	NICE_EXIT	000004F6 R	05	
CIRC_NAME	000001A5 R	03	NICE_MESSAGE	0000019E R	03	
CIR_CNT_BADM	= 00000004		NICE_MSG	0000018B R	03	
CIR_CNT_BADV	= 00000002		NICE_ROUTINE	00000324 R	05	
CNTRLCMSG	0000004C R	02	NICE_SIZE	= 00000005		
CNTR_TBL	000001A3 R	02	NMASC_CTCIR_ACL	= 00000322		
CONTROL_CM	= 00000002		NMASC_CTCIR_CRL	= 00000325		
CONTROL_CV	= 00000001		NMASC_CTCIR_DEI	= 000003FC		
COUNTER	000001D7 R	03	NMASC_CTCIR_DEO	= 000003FD		
COUNTER_MSG	0000011C R	02	NMASC_CTCIR_IFL	= 00000335		
DCS_TERM	***** X	05	NMASC_CTCIR_LBE	= 00000411		
DEV	0000004B R	03	NMASC_CTCIR_LDN	= 00000334		
DEVBUF	00000053 R	03	NMASC_CTCIR_LIR	= 000004D8		
DIBSB_DEVCLASS	= 00000004		NMASC_CTCIR_LPE	= 0000044D		
DIBSK_LENGTH	= 00000074		NMASC_CTCIR_LRT	= 00000407		
END_ADR	000001E7 R	03	NMASC_CTCIR_NIR	= 000004DA		
ERROR_COUNT	0000015B R	03	NMASC_CTCIR_RBE	= 00000410		
ERROR_EXIT	000006B3 R	05	NMASC_CTCIR_RIR	= 000004D9		
ERR_MSG_CTR	000000E3 R	02	NMASC_CTCIR_RPE	= 0000044C		
EXIT_DESC	00000167 R	03	NMASC_CTCIR_RRT	= 00000406		
EXIT_HANDLER	0000071C R	05	NMASC_CTCIR_SLT	= 0000041B		
FABSB_BID	= 00000000		NMASC_CTCIR_TCL	= 0000032C		
FABSB_FNS	= 00000034		NMASC_CTNOD_APL	= 00000384		
FABSC_BID	= 00000003		NMASC_CTNOD_NOL	= 00000386		
FABSC_BLN	= 00000050		NMASC_CTNOD_NUL	= 00000385		
FABSC_SEQ	= 00000000		NMASC_CTNOD_OPL	= 00000387		
FABSC_VAR	= 00000002		NMASC_CTNOD_PFE	= 0000038E		
FABSL_ALQ	= 00000010		NMASC_CTNOD_RSE	= 00000280		

UETNETS00
Symbol table

K 16
VAX/VMS UETP checker for DECnet counters 16-SEP-1984 01:29:03 VAX/VMS Macro V04-00
5-SEP-1984 04:25:57 [UETP.SRC]UETNETS00.MAR;1

Page 30
(14)

```

NMASC_CTNOB_RTO      = 00000276
NMASC_CTNOB_RUL      = 00000398
NMASC_CTNOB_VER      = 000003A2
NMASC_ENT_CTR        = 00000003
NMASC_ENT_NOD        = 00000000
NMASC_FNC_REA        = 00000014
NMASC_OPINF_COU      = 00000003
NMASC_STS_DON        = FFFFFFF80
NMASC_STS_MOR        = 00000002
NMASC_STS_SUC        = 00000001
NMASS_CNT_TYP        = 0000000C
NMASS_CNT_WID        = 00000002
NMAV_CNT_MAP         = 0000000C
NMAV_CNT_TYP         = 00000000
NMAV_CNT_WID         = 0000000D
NMAV_OPT_INF         = 00000004
NML$INITIALIZE       ***** X 05
NML$PROCESS_NICE     ***** X 05
NML$TERMINATE        ***** X 05
NMLINIT_ERR          = 000000A8 R 02
NOD                  = 00000000
NODE                 = 0000013E R 02
NODE_ADR             = 000001B9 R 03
NODE_ADR_DESC        = 00000183 R 03
NODE_NAME            = 000001AF R 03
NODE_WRD             = 000001A1 R 03
NOD_CNT_BADM         = 00000008
NOD_CNT_BADV         = 00000003
NO RMS_AST_TABLE     = 00000020 R 02
NRAT_LENGTH          = 00000014
OTSS$VT_TI_L         ***** X 05
PC1...               = 0000023F R 02
PC2...               = 00000481 R 02
RAB$B_RAC            = 0000001E
RAB$C_BID            = 00000001
RAB$C_BLN            = 00000044
RAB$C_SEQ            = 00000000
RAB$C_CTX            = 00000018
RAB$C_FAB            = 0000003C
RAB$C_RBF            = 00000028
RAB$C_ROP            = 00000004
RAB$C_STS            = 00000008
RAB$C_STV            = 0000000C
RAB$W_RSZ            = 00000022
RECORD               = 00000079 R 02
RMSS$ FACILITY       = 00000001
RMSS$ BLN            ***** X 02
RMSS$ BUSY           ***** X 02
RMSS$ CDA            ***** X 02
RMSS$ FAB            ***** X 02
RMSS$ FACILITY       ***** X 05
RMSS$ RAB            ***** X 02
RMS_COMMON           = 0000063E R 05
RMS_ERROR            = 00000608 R 05
RMS_ERR_STRING       = 00000087 R 02
SHRS_ABENDD          = 000010E0
SHRS_BEIND           = 00001038

```

```

SHRS_ENDEDD          = 00001080
SHRS_OPENIN          = 00001098
SHRS_TEXT            = 00001130
SHRT_RPRTM           = 00000001
SHRT_RPRTV           = 00000000
SS$_BADPARAM         ***** X 05
SS$_CONTROL          ***** X 05
SS$_NORMAL           ***** X 05
SS$_NOTRAN           ***** X 05
SS$_SSFAL            ***** X 05
SS$_WASSET           ***** X 05
SSERROR              = 000004F7 R 05
STATUS               = 0000015F R 03
STSS$ ERROR          = 00000002
STSS$ INFO           = 00000003
STSS$ SUCCESS        = 00000001
STSS$ WARNING        = 00000000
STSSM_INHIB MSG      = 10000000
STSS$ FAC NO         = 0000000C
STSS$ SEVERITY       = 00000003
STSSV_FAC NO         = 00000010
STSSV_SEVERITY       = 00000000
SUC_EXIT             = 000002F4 R 05
SYSS$ ASSIGN         ***** GX 05
SYSS$ CLOSE          ***** GX 05
SYSS$ CONNECT        ***** GX 05
SYSS$ CREATE         ***** GX 05
SYSS$ DCLEXH         ***** GX 05
SYSS$ DISCONNECT     ***** GX 05
SYSS$ EXIT           ***** GX 05
SYSS$ FAQ            ***** X 05
SYSS$ GET            ***** GX 05
SYSS$ GETDEV         ***** GX 05
SYSS$ GETMSG         ***** GX 05
SYSS$ OPEN           ***** GX 05
SYSS$ PUT            ***** GX 05
SYSS$ PUTMSG         ***** GX 05
SYSS$ QIOW           ***** GX 05
SYSS$ SETAST         ***** GX 05
SYSS$ SETPRN         ***** GX 05
SYSS$ SETSFM         ***** GX 05
SYSS$ TRNLOG         ***** GX 05
TBL_END              = 0000023F R 02
TBL_SIZE             = 0000001A
TEXT_BUFFER          = 00000084
THRU                 = 0000014F R 02
TO                   = 0000014C R 02
TSTNAM               = 0000000F R 02
TTCHAN               = 00000000 R 03
TTNAME               = 0000000A R 03
TTNAME_LEN           = 0000000B
TTNAME_ROPTR         = 00000040 R 02
TTNAME_RWPTR         = 000000C2 R 03
TYPE                 = 000001DB R 03
TYPE1                = 000001DF R 03
TYPE2                = 000001E3 R 03
UETNETS00            = 00000000 RG 05

```


UETNETS00
Symbol table

VAX/VMS UETP checker for DECnet counters L 16
16-SEP-1984 01:29:03 VAX/VMS Macro V04-00
5-SEP-1984 04:25:57 [UETP.SRC]UETNETS00.MAR;1

Page 31
(14)

UETP = 00740000
UETPS_ABENDD = 007410E0
UETPS_ABORTC = 0074832B
UETPS_BEGINDD = 00741038
UETPS_ENDEDD = 00741080
UETPS_ERBOXPROC = 00748C20
UETPS_FACILITY = 00000074
UETPS_OPENIN = 00741098
UETPS_TEXT = 00741130
ZERO = 0000019F R 02

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000481 (1153.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000314 (788.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
\$RMSNAM	0000001A (26.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
UETNETS00	00000768 (1896.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.12	00:00:00.92
Command processing	136	00:00:00.72	00:00:03.26
Pass 1	490	00:00:18.77	00:00:40.96
Symbol table sort	0	00:00:02.12	00:00:03.31
Pass 2	204	00:00:04.31	00:00:08.88
Symbol table output	29	00:00:00.22	00:00:00.55
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	890	00:00:26.29	00:00:57.91

The working set limit was 2000 pages.
103700 bytes (203 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1519 non-local and 41 local symbols.
1095 source lines were read in Pass 1, producing 34 object records in Pass 2.
50 pages of virtual memory were used to define 43 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	1
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	37
TOTALS (all libraries)	39

1809 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:UETNETS00/OBJ=OBJ\$:UETNETS00 MSRC\$:UETNETS00/UPDATE=(ENH\$:UETNETS00)+EXECMLS/LIB+LIB\$:UETP/LIB

0411 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY